**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Рекурсия**

**Вариант 6**

Студент гр. 8304 _____ Щука А.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2019

**Цель работы.**

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++ в фреймворке Qt.

**Постановка задачи.**

1) проанализировать полученное задание, выделив рекурсивно определяемые информационные объекты и (или) действия;
2) разработать программу, использующую рекурсию;
3) сопоставить рекурсивное решение с итеративным решением задачи;
4) сделать вывод о целесообразности и эффективности рекурсивного решения данной задачи.

Вариант 6: Построить синтаксический анализатор для понятия выражение.
простое_выражение::=простой_идентификатор |
(простое_выражение знак_операции простое_выражение)
простой_идентификатор::= буква
знак_операции:: = – | + | *

**Описание алгоритма.**

Для вычисления значения выражения методом рекурсии необходимо считать строку, затем рекурсивно проверить корректность выражения.
При проверке на соответствие определению «выражение», проверяется либо наличие буквы, либо наличие конструкции «(выражение знак выражение)». Алгоритм возвращает false если строка не соответствует определению, и true в ином случае. После завершение алгоритма итератор должен находиться в конце строки, в противном случае строка неккореткна.

```
проверить_выражение(позиция)
        если проверить_букву(позиция) = true
                вернуть true

        если текущий_элемент = ”(”
                позиция += 1
```

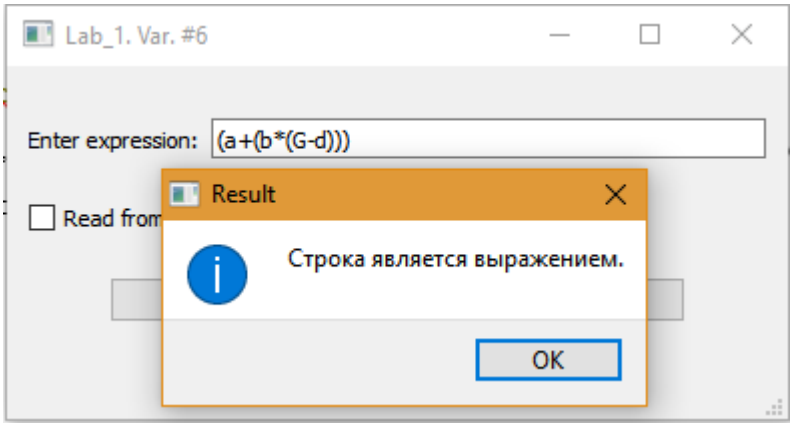
```
иначе
        вернуть false
если проверить_выражение(позиция) = true
        позиция += 1
иначе
        вернуть false
если проверить_знак(позиция) = true
        позиция += 1
иначе
        вернуть false
если проверить_выражение(позиция) = true
        позиция += 1
иначе
        вернуть false
если текущий_элемент != ”)”
        вернуть false
вернуть true
```

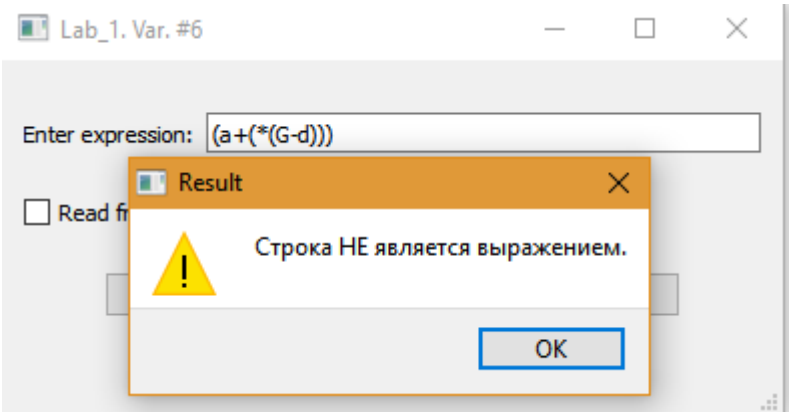
**Спецификация программы.**

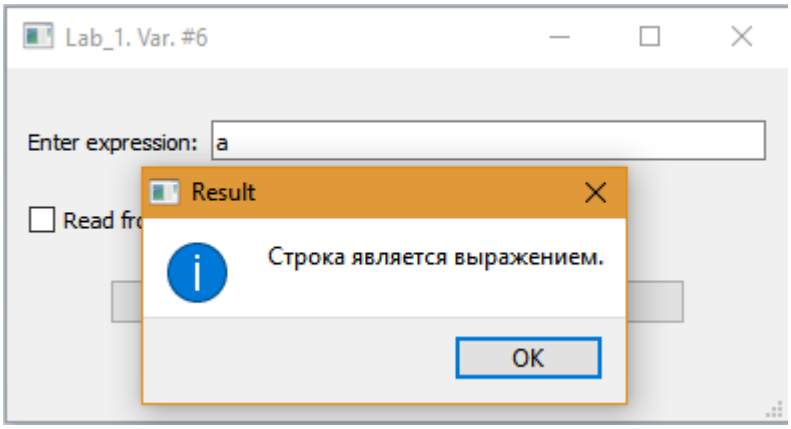
Программа предназначена для синтаксического анализа выражения методом рекурсии.

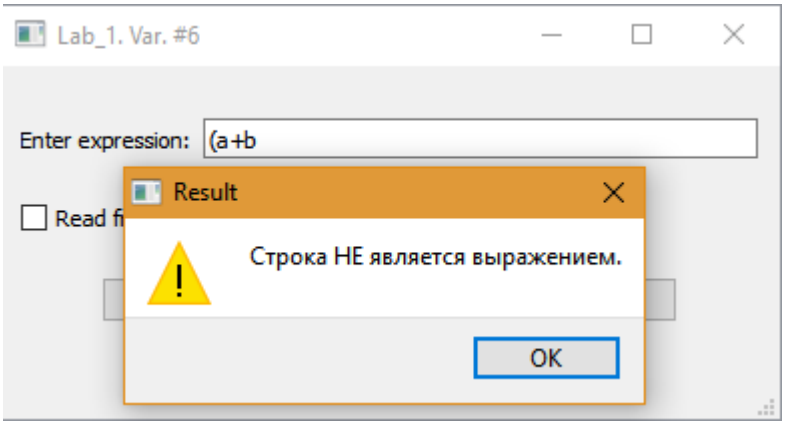
Программа написана на языке C++. Входными данными является строка. Выходными данными являются промежуточные значения проверки выражения (записываются в qDebug()) и конечный результат (корректна строка или нет). Данные выводятся в QMessageBox.

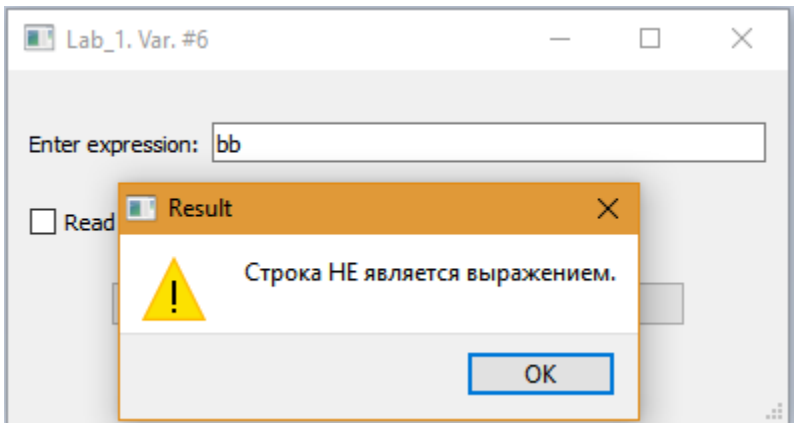
**Тестирование.**

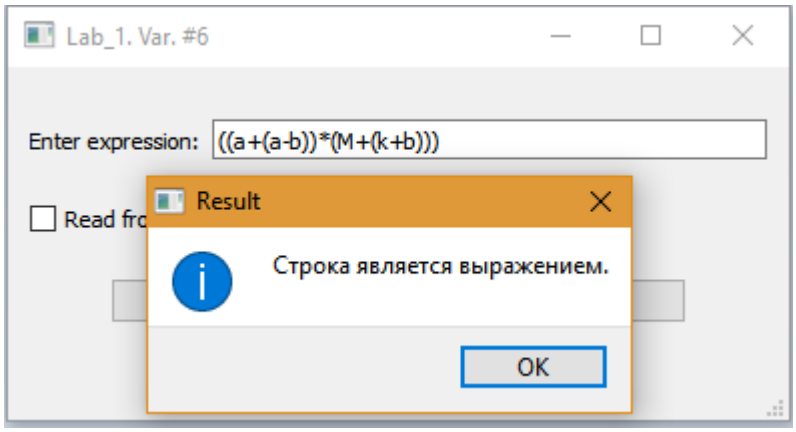





**Анализ алгоритма.**

Для экономии памяти передается итератор по строке, что позволяет избежать копирования. Алгоритм работает за линейное время. Недостаток рекурсивного алгоритма – ограниченный стек вызовов функций, что в свою очередь накладывает ограничение на размер входной строки, а также затраты производительности на вызов функций.

**Описание функций и СД**

Функции, реализующие интерфейс синтаксического анализатора.

```
bool isExpression(std::string::iterator& pos, size_t depth = 0) const;
```

Метод класса, который принимает на вход итератор и глубину рекурсии (для отладки). Возвращаемое значение: true если выражение корректно, false – если нет. Реализует алгоритм синтаксической проверки выражения.

```
bool isSign(const char ch) const;
```

Метод класса, который принимает на вход символ, и возвращает true, если символ = “знак” или false в ином случае.

bool **isLetter**(const char ch) const;

Метод класса, который принимает на вход символ, и возвращает true, если символ = “буква” или false в ином случае.

## Выводы.

В ходе работы я познакомился с рекурсией, научился работать с данным методом. Я считаю, что метод рекурсии для данной задачи проверки выражения на корректность не эффективен.

## Приложение А.

## mainwondow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QDebug>
#include <QString>
#include <QFile>
#include <QMessageBox>
#include <QFileDialog>
#include <QTextStream>


namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private:
    bool isSign(const char ch) const;

    bool isLetter(const char ch) const;

    bool isExpression(std::string::iterator& pos,
                      size_t depth = 0) const;

private slots:
    void on_check_pushButton_clicked();
```

```
    void on_checkBox_clicked();

public slots:
    void test();

private:
    Ui::MainWindow *ui;
    std::string result;
    QTextStream* in;
    QFile* file;

};

#endif // MAINWINDOW_H
```

## main.cpp

```
#include "mainwindow.h"
#include <QApplication>


//var #6

int main(int argc, char *argv[]) {
    setlocale(LC_ALL, "Russian");

    QApplication a(argc, argv);

    MainWindow window;
    if (argc == 2 && strcmp(argv[1], "--test") == 0) {
        window.test();
        return 0;
    }

    window.setWindowTitle("Lab_1. Var. #6");
    window.show();

    return a.exec();
}
```

## mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"


MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    in = new QTextStream;
    file = new QFile;
}


MainWindow::~MainWindow()
{
    delete ui;
    delete in;
    delete file;
}


void MainWindow::on_check_pushButton_clicked()
{
    /*
     * При нажатии на кнопку "Check expression" происходит проверка выражения
     * с помощью рекурсивного алгоритма, реализованного в классе
     * SyntaxAnalysis
    */

    qDebug() << "Анализатор выражения.";

    if (ui->checkBox->isChecked()) {
        result = in->readAll().toStdString();
    }
    else {
```

```
        result = ui->lineEdit->text().toStdString();
    }

    qDebug() << "Считанная строка: " << result.c_str();

    auto it_begin = result.begin();
    auto it_end = result.end();

    if (isExpression(it_begin) && ++it_begin == it_end) {
        qDebug() << "Строка является выражением.";
        QMessageBox::information(this, "Result", "Строка является выражением.");
    }
    else {
        qDebug() << "Строка НЕ является выражением.";
        QMessageBox::warning(this, "Result", "Строка НЕ является выражением.");
    }

    ui->checkBox->setChecked(false);
    ui->label->setEnabled(true);
    ui->lineEdit->setEnabled(true);
}


void MainWindow::on_checkBox_clicked()
{

    /*
     * Переключение между считыванием из строки и из файла
    */

    if (ui->checkBox->isChecked()) {
        ui->label->setEnabled(false);
        ui->lineEdit->setEnabled(false);
        file->setFileName(QFileDialog::getOpenFileName(this,
                                                       "Open file"));
        file->open(QFile::ReadOnly | QFile::Text);
        in->setDevice(file);
    }
    else {
        ui->label->setEnabled(true);
        ui->lineEdit->setEnabled(true);
    }
}


bool MainWindow::isExpression(std::string::iterator& pos,
                              size_t depth) const
{
    std::string debugString = "";
    for (size_t i = 0; i < depth; ++i)
        debugString += " ";
    qDebug() << debugString.c_str() << __func__ << "глубина рекурсии:" << depth;

    if (isLetter(*pos)) {
        qDebug() << debugString.c_str() << "Выражение корректно";
        return true;
    }

    if (*pos == '(') {
        ++pos;
    }
    else {
        qDebug() << debugString.c_str() << "отсутствует '(' или 'идентификатор'";
        return false;
    }

    if (isExpression(pos, depth+1)) {
        ++pos;
    }
    else {
        qDebug() << debugString.c_str() << "отсутвует 'выражение'";
        return false;
    }

    if (isSign(*pos)) {
        ++pos;
    }
    else {
        qDebug() << debugString.c_str() << "отсутствует 'знак'";
```

```
        return false;
    }

    if (isExpression(pos, depth+1)) {
        ++pos;
    }
    else {
        qDebug() << debugString.c_str() << "отсутствует 'выражение'";
        return false;
    }

    if (*pos != ')') {
        qDebug() << debugString.c_str() << "отсутствует ')'";
        return false;
    }

    qDebug() << debugString.c_str() << "Выражение корректно";
    return true;
}


bool MainWindow::isLetter(const char ch) const
{
    //проверка символа на соответствие букве
    return ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'));
}


bool MainWindow::isSign(const char ch) const
{
    //проверка символа на соответствие знаку
    return (ch == '*' || ch == '+' || ch == '-');
}


void MainWindow::test()
{
    file->setFileName(QDir::currentPath() + "/Tests/test.txt");
    file->open(QFile::ReadOnly | QFile::Text);
    in->setDevice(file);

    std::string result;
    while ((result = in->readLine().toStdString()) != "") {
        auto it_begin = result.begin();
        auto it_end = result.end();

        qDebug();
        qDebug() << "Считанная строка:" << result.c_str();
        if (isExpression(it_begin) && ++it_begin == it_end) {
            qDebug() << "Строка является выражением.";
        }
        else {
            qDebug() << "Строка НЕ является выражением.";
        }
        qDebug();
    }
}
```

## mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>390</width>
    <height>176</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <layout class="QVBoxLayout" name="verticalLayout">
    <item>
     <layout class="QHBoxLayout" name="horizontalLayout">
```

```
    <item>
     <widget class="QLabel" name="label">
      <property name="text">
       <string>Enter expression:</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QLineEdit" name="lineEdit"/>
    </item>
   </layout>
  </item>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_2">
    <item>
     <widget class="QCheckBox" name="checkBox">
      <property name="text">
       <string>Read from file</string>
      </property>
     </widget>
    </item>
   </layout>
  </item>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_3">
    <item>
     <spacer name="horizontalSpacer_2">
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
      <property name="sizeType">
       <enum>QSizePolicy::Preferred</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>40</width>
        <height>20</height>
       </size>
      </property>
     </spacer>
    </item>
    <item>
     <widget class="QPushButton" name="check_pushButton">
      <property name="text">
       <string>Check expression</string>
      </property>
     </widget>
    </item>
    <item>
     <spacer name="horizontalSpacer">
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
      <property name="sizeType">
       <enum>QSizePolicy::Preferred</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>40</width>
        <height>20</height>
       </size>
      </property>
     </spacer>
    </item>
   </layout>
  </item>
 </layout>
</widget>
<widget class="QMenuBar" name="menuBar">
 <property name="geometry">
  <rect>
   <x>0</x>
   <y>0</y>
   <width>390</width>
   <height>21</height>
  </rect>
 </property>
</widget>
<widget class="QStatusBar" name="statusBar"/>
```

```
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```